Patent Abstract

GER 1995-10-05 4410775 **Controller and operating system for this controller**
ANNOTATED TITLE- Steuergeroat und Betriebssystem foOr dieses Steuergeroat

INVENTOR- NACHTRo GLICH

APPLICANT- Siemens AG 80333 MoOnchen DE
APPLICANT- Bayerische Motoren Werke AG 80809 MoOnchen DE
APPLICANT- Daimler-Benz Aktiengesellschaft 70567 Stuttgart DE
PATENT NUMBER- 04410775/DE-A1
PATENT APPLICATION NUMBER- 04410775
DATE FILED- 1994-03-28
DOCUMENT TYPE- A1, DOCUMENT LAID OPEN (FIRST PUBLICATION)
PUBLICATION DATE- 1995-10-05
INTERNATIONAL PATENT CLASS- G05B019042; F02D04126; G05B019042
PATENT APPLICATION PRIORITY- 4410775, A
PRIORITY COUNTRY CODE- DE, Germany, Ged. Rep. of
PRIORITY DATE- 1994-03-28
FILING LANGUAGE- German
LANGUAGE- German NDN- 203-0337-8947-0

The controller serves in particular for steering functions in motor; vehicles (motor control, transmission price increase etc.). It; exhibits a processor (1), which is provided with RAM and read-only; memories (3, 4). Working off trials or tasks is coordinated temporally; and steered by the operating system. It is both for the treatment of; tasks according to a batch processing method (non preemptive; scheduling), with which processing a current task not be interrupted; cannot, and for the treatment of tasks according to a displacement; method (preemptive scheduling), with which processing a task for; working off another task be interrupted can, furnished. The kind of; processing is attached to each task as attribute, by which the; respective method of handling is specified.
EXEMPLARY CLAIMS- 1. Controller (1), in particular to steering functions in motor vehicles, that with a processor (2), with data memories (3, 4), also-and expenditure circuits (6) as well as with an operating system provided is characterized, by which the processing of tasks is coordinated temporally and steered, by it, that the operating system is furnished both for the treatment of tasks according to a batch processing method (nonpreemptive 5 scheduling), with which processing a task not be interrupted cannot and for the treatment of tasks according to a displacement method (preemptive scheduling), with that the processing of a task for 10 processing another task are interrupted can, and 2. Operating system for a controller (1), to steering functions in motor vehicles, by which the processing of tasks is coordinated temporally and steered by the controller, by it are in particular marked, that tasks worked on both according to a batch processing method (nonpreemptive scheduling), with which processing a task not be interrupted cannot and according to a displacement method, with 25 that processing a task for processing another task be interrupted can (preemptive scheduling), and that the kind of the processing each task as attribute one assigns, by which jewei-the 30 lige processing method is specified.

NO-DESCRIPTORS

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 44 10 775 A1**

(51) Int. Cl.[6]:
G 05 B 19/042
F 02 D 41/26

(21) Aktenzeichen: P 44 10 775.7
(22) Anmeldetag: 28. 3. 94
(43) Offenlegungstag: 5. 10. 95

DE 44 10 775 A1

(71) Anmelder:
Siemens AG, 80333 München, DE; Bayerische Motoren Werke AG, 80809 München, DE; Daimler-Benz Aktiengesellschaft, 70567 Stuttgart, DE

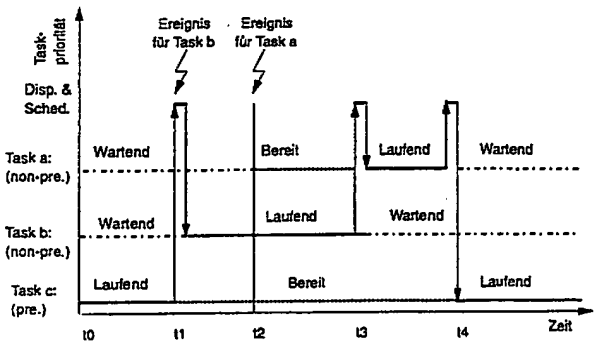
(74) Vertreter:
Fuchs, F., Dr.-Ing., Pat.-Anw., 81541 München

(72) Erfinder:
Erfinder wird später genannt werden

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Steuergerät und Betriebssystem für dieses Steuergerät

(57) Das Steuergerät dient insbesondere zum Steuern von Funktionen in Kraftfahrzeugen (Motorsteuerung, Getriebesteuerung usw.). Es weist einen Prozessor (1) auf, der mit RAM- und Festwertspeichern (3, 4) versehen ist. Durch das Betriebssystem wird das Abarbeiten von Prozessen oder Tasks zeitlich koordiniert und gesteuert. Es ist sowohl für die Bearbeitung von Tasks nach einer Stapelverarbeitungsmethode (non-preemptive scheduling), bei der die Abarbeitung einer laufenden Task nicht unterbrochen werden kann, als auch für die Bearbeitung von Tasks nach einer Verdrängungsmethode (preemptive scheduling), bei der die Abarbeitung einer Task zum Abarbeiten einer anderen Task unterbrochen werden kann, eingerichtet. Die Art der Abarbeitung wird jeder Task als Attribut zugeordnet, durch das die jeweilige Bearbeitungsmethode festgelegt wird.



DE 44 10 775 A1

Beschreibung

Die Erfindung betrifft ein Steuergerät nach dem Oberbegriff von Patentanspruch 1 und ein Betriebssystem nach dem Oberbegriff von Anspruch 2.

Bei einer bekannten Motorsteuerung (DE 38 26 526 A1) laufen Programme unterschiedlicher zeitlicher Priorität ab. Dabei sind Programme prioritätshoch, mit denen bei einem vorgegebenen Kurbelwinkel ein Motorsteuervorgang ausgelöst wird, z. B. Kraftstoff eingespritzt oder das Kraftstoff-Luft-Gemisch gezündet wird. Mit niedriger Priorität laufen sogenannte Hintergrundprogramme ab.

Bei dem Entwurf von Steuerungen wird die von dem Steuergerät zu erledigende Steuerungsaufgabe üblicherweise in Teilaufgaben gegliedert und jede Teilaufgabe als ein in sich abgeschlossenes Programm realisiert. Ein solches Programm wird als Task — oder auch als Prozeß oder Teilaufgabe — bezeichnet. Betriebssysteme die mehrere Tasks verwalten können, werden als Multitasking-Betriebssysteme bezeichnet.

Bei der Durchführung der Steuerungsaufgaben tritt häufig eine Situation auf, bei der sich mehrere Tasks um die Benutzung der zentralen Rechnereinheit, der CPU, bemühen. Die Aufgabe des Multitasking-Betriebssystems besteht darin, der CPU immer nur eine von mehreren Tasks zuzuteilen. Der Teil des Betriebssystems, der die Taskwechsel vollzieht, wird als Dispatcher bezeichnet. Derjenige Teil des Betriebssystems, der entscheidet, welcher Task die CPU zugeteilt wird, wird als Scheduler bezeichnet.

Ein Taskwechsel kann von dem Betriebssystem grundsätzlich auf zwei verschiedene Arten durchgeführt werden:

— nach einer Verdrängungsmethode — im folgenden als "preemptive scheduling" bezeichnet —, bei der die Abarbeitung einer Task von dem Betriebssystem unterbrochen und die CPU einer anderen wartenden Task zugeteilt werden kann, und
— nach einer Stapelverarbeitungsmethode — im folgenden: "non-preemptive scheduling" —, bei der eine laufende, d. h. derzeitig abgearbeitete Task nicht durch eine andere Task verdrängt werden kann.

Bei dem preemptive scheduling benötigt das Betriebssystem zusätzliche Rechnerzeit und Arbeitsspeicherkapazität, um die Daten der Task zu sichern, die unterbrochen wird und später weiter bearbeitet werden muß. Bei dem non-preemptive scheduling muß beim Entwickeln der Programme für das Steuergerät darauf geachtet werden, daß die Laufzeiten der einzelnen Tasks nicht die Echtzeitbedingungen des Systems verletzen. Es kann deshalb erforderlich sein, daß eine zusammenhängende Teilaufgabe in mehrere Tasks aufgeteilt werden muß.

Beim Entwickeln und Programmieren von Steuergeräten muß man sich bislang für eine der Betriebssystemarten entscheiden. Entweder entscheidet sich der Entwickler für ein non-preemptive Betriebssystem, für das kein zusätzlicher Arbeitsspeicherplatz erforderlich ist, der Aufwand für die Programmerstellung aber höher ist, oder er entscheidet sich für ein preemptive Betriebssystem, das zwar zusätzlichen Speicher benötigt, die Programmerstellung aber vereinfacht.

Der Erfindung liegt die Aufgabe zugrunde, ein Steuergerät und ein Betriebssystem für ein solches Steuergerät zu schaffen, das den Aufwand für Hardware und Software verringert und die Verarbeitungskapazität des Steuergeräts möglichst gut ausnutzt.

Diese Aufgabe wird erfindungsgemäß durch das Steuergerät nach Anspruch 1 und das Betriebssystem nach Anspruch 2 gelöst.

Ein Ausführungsbeispiel der Erfindung wird im folgenden anhand der Zeichnung erläutert. Es zeigen:

Fig. 1 ein Steuergerät gemäß der Erfindung in Blockdiagrammdarstellung,

Fig. 2 die verschiedenen Zustände, die eine in dem Steuergerät nach **Fig.** 1 abzuarbeitende Task einnehmen kann,

Fig. 3 eine Taskfolge bei einem Betriebssystem mit preemptive scheduling,

Fig. 4 eine Taskfolge bei einem Betriebssystem mit non-preemptive scheduling, und

Fig. 5 eine Taskfolge bei einem erfindungsgemäßen Betriebssystem mit mixed-preemptive scheduling.

Ein Steuergerät **1** (**Fig.** 1) weist einen Zentralprozessor oder CPU **2**, einen Arbeitsspeicher oder RAM **3**, einen Festwertspeicher oder EPROM **4** sowie Ein- und Ausgabeschaltungen oder I/O-Ports **6** auf. Diese Schaltungsbestandteile sind durch Daten- und Signalleitungen **7** bis **10** und ein Bussystem **11** untereinander verbunden. An den I/O-Port **6** sind eine Eingangsleitung **12** und Ausgangsleitung **13** angeschlossen, über die das Steuergerät **1** mit dem zu steuernden System, z. B. mit der Einspritzanlage, der Zündsteuerung, der Getriebesteuerung usw. des Kraftfahrzeugs verbunden ist. Über die Eingangsleitung **12** gelangen die von hier nicht dargestellten Sensoren gelieferten Meßwerte zu dem Steuergerät, über die Leitung **13** werden von diesem Steuersignale an eine ebenfalls nicht dargestellte Einrichtung, z. B. eine Motorsteuerung, ausgegeben.

Eine in dem Steuergerät **1** abzuarbeitende Task kann vier verschiedene Zustände (Taskzustände) einnehmen, die aus **Fig.** 2 ersichtlich sind:

— Ruhend: Die Task belegt keine Betriebsmittel; sie bleibt in diesem Zustand, bis sie von einer anderen Task oder dem Betriebssystem gestartet wird.
— Bereit: Die Task bewirbt sich um die Zuteilung der CPU **2**.
— Laufend (oder Aktiv): Die Task hat die CPU **2** zugeteilt bekommen und wird von dieser abgearbeitet.
— Wartend: Die Task wartet auf ein Ereignis, z. B. auf das Ergebnis einer anderen Task. Tritt dieses Ereignis ein, so wird die Task freigegeben und gelangt in den Zustand Bereit.

Der Taskwechselmechanismus kann nach einer der eingangs angegebenen Methoden, der Stapelverarbeitung oder der Verdrängungsmethode, erfolgen.

Bei der Verdrängungsmethode oder preemptive scheduling kann eine laufende Task (z. B. durch einen Interrupt) von dem Betriebssystem unterbrochen und die CPU **2** einer anderen wartenden Task zugeteilt werden, wie nun anhand von **Fig.** 3 erläutert wird.

In dem dargestellten Beispiel werden drei Tasks a, b und c betrachtet, wobei die Task a eine höhere Priorität als die Tasks b und c und die Task b eine höhere Priorität als die Task c hat. Zu einem Zeitpunkt t_0 warten die höherprioren Task a und b auf ein Ereignis, während die Task c läuft, d. h. abgearbeitet wird.

Zu einem Zeitpunkt t_1 tritt ein Ereignis — ein Interrupt — für die Task b ein. Der Dispatcher und der

Scheduler werden gestartet. Da außer der Task c eine höherpriore Task bereit ist, wird die Task c unterbrochen und ihr Taskkontext gerettet.

Unter dem Taskkontext wird hier die Datengesamtheit verstanden, die erforderlich ist, um die unterbrochene Task später fortzusetzen. Im wesentlichen besteht der Taskkontext aus dem Inhalt der — hier nicht dargestellten, da allgemein bekannt — Register der CPU 2.

Die Task b wird gestartet.

Zu einem Zeitpunkt t_2 tritt ein Ereignis (Interrupt) für die Task a ein. Der Dispatcher und der Scheduler werden gestartet. Da neben der Tasks b und c die höherpriore Task a bereit ist, wird die Task b unterbrochen und ihr Taskkontext gerettet. Die Task a wird gestartet.

Zu einem Zeitpunkt t_3 ist die Task a beendet. Der Dispatcher und der Scheduler werden gestartet. Von den dann bereiten Tasks hat die Task b die höchste Priorität. Da diese unterbrochen worden ist, restauriert der Dispatcher ihren Kontext. Anschließend kann die Task fortgesetzt werden, d. h. ihre noch nicht abgearbeiteten Befehle werden nun abgearbeitet.

Zu einem Zeitpunkt t_4 ist die Task b beendet. Der Dispatcher und der Scheduler werden gestartet. Zu diesem Zeitpunkt ist nur noch die Task c bereit. Da diese unterbrochen worden war, restauriert der Dispatcher ihren Kontext. Anschließend kann die Task b mit ihrer Befehlsfolge weiter durchgeführt werden.

Da die Tasks mitten in ihrer Befehlsfolge von höherprioren Tasks unterbrochen werden können, muß im Falle einer Unterbrechung der Dispatcher den Taskkontext retten. Dazu wird der Taskkontext temporär in einem Bereich des RAM-Speichers 3 zwischengespeichert. Wenn nach einem Taskwechsel die unterbrochene Task die CPU wieder zugeteilt bekommt, wie z. B. die Task b zu dem Zeitpunkt t_3, muß der Dispatcher zuerst den Taskkontext wieder restaurieren.

Da die Tasks hier von "wichtigeren" Tasks verdrängt werden können, braucht bei der Entwicklung des Programms die Länge einer Task nicht berücksichtigt zu werden. Da in dem Beispiel die Task c von höherprioren Task unterbrochen werden kann, kann diese Task beliebig lang sein, ohne daß Echtzeitverhalten für höherpriorer Tasks und damit das Echtzeitverhalten des Steuergeräts zu gefährden. Allerdings benötigt das Steuergerät für die Rettung des Taskkontextes wie erwähnt zusätzliche Rechenzeit und RAM-Speicherplatz, und zwar in Abhängigkeit von dem Umfang des Kontextes und der Gesamtzahl der Tasks.

Bei der non-preemptive scheduling oder Stapelverarbeitungsmethode kann eine laufende Task nicht durch eine andere Task verdrängt, d. h. unterbrochen werden. Dies ist in **Fig.** 4 dargestellt.

Auch in diesem Beispiel stehen drei Tasks a, b und c an, von denen die Task a eine höhere Priorität als die Tasks b und c und die Task b eine höhere Priorität als die Task c hat. Zu einem Zeitpunkt t_0 warten die höherprioren Tasks a und b auf ein Ereignis, während die Task c läuft, d. h. abgearbeitet wird.

Zu einem Zeitpunkt t_1 tritt ein Ereignis (Interrupt) für die Task b ein. Die Task b gelangt deshalb in den Zustand Bereit. Da laufende Tasks nicht unterbrochen werden können, muß die Task b bis zu einem Zeitpunkt t_2 warten, zu dem die Task c beendet worden ist und die Task b gestartet werden kann.

Zu einem Zeitpunkt t_3 tritt ein Ereignis (Interrupt) für die Task a ein. Die Task gelangt deshalb in den Zustand Bereit. Da laufende Tasks nicht unterbrochen werden können, muß die Task a bis zu einem Zeitpunkt t_4 warten. Zu diesem Zeitpunkt ist die Task b beendet und die Task a kann gestartet werden.

Da die Tasks hier nicht von höherprioren Tasks unterbrochen werden können, muß der Dispatcher den Taskkontext weder retten noch restaurieren. Es wird kein Speicherplatz in dem RAM-Speicher 3 belegt. Da die Tasks nicht unterbrochen werden können, muß andererseits der Entwickler des Programms darauf achten, daß die Laufzeit einer Task nicht die Echtzeitbedingungen verletzt. Dabei kann zusätzlicher Programmieraufwand erforderlich werden.

Die Art des Taskwechselmechanismus wird deshalb hier einer Task als Attribut "preemptive scheduling" oder "non-preemptive scheduling" zugeordnet. Damit ergibt sich eine Abarbeitung der Taskfolge nach einem "mixed-preemptive scheduling", das aus **Fig.** 5 ersichtlich ist. Auch hier liegen drei Tasks a, b und c vor, wobei die Task a eine höhere Priorität als die Tasks b und c und die Task b eine höhere Priorität als die Task c haben. Die Task c ist als eine preemptive Task und die Tasks a und b als eine non-preemptive Tasks deklariert.

Zu einem Zeitpunkt t_0 warten die höherprioren Tasks a und b auf ein Ereignis; die Task c läuft.

Zu einem Zeitpunkt t_1 tritt ein Ereignis (Interrupt) für die Task b ein. Der Dispatcher und der Scheduler des Betriebssystems werden gestartet. Da neben der Task c eine höherpriore Task bereit ist und außerdem die Task c als eine preemptive Task deklariert ist, wird die Task c unterbrochen und ihr Taskkontext gerettet. Die Task b wird gestartet.

Zu einem Zeitpunkt t_2 tritt ein Ereignis (Interrupt) für die Task a ein. Die Task gelangt deshalb in den Zustand bereit. Da die laufende Task b als non-preemptive deklariert ist, kann sie nicht unterbrochen werden: Die Task a muß bis zu einem Zeitpunkt t3 warten, zu dem die Task b beendet worden ist und die Task a gestartet werden kann.

Zu einem Zeitpunkt t_4 ist die Task a beendet. Der Dispatcher und der Scheduler des Betriebssystems werden gestartet. Zu diesem Zeitpunkt ist nur noch die Task c bereit. Da diese unterbrochen wurde, restauriert der Dispatcher ihren Kontext. Anschließend kann die Task mit ihrer Befehlsfolge weiter verarbeitet werden. Da als "preemptive" deklarierte Tasks mitten in der Befehlsfolge von höherprioren Tasks unterbrochen werden können, muß der Dispatcher nur für diese Tasks den Taskkontext, d. h. z. B. den Inhalt der CPU-Register retten.

Das hier beschriebene mixed-preemptive scheduling oder kombinierte Multitasking-Betriebssystem ermöglicht es, bei der Entwicklung eines Steuerungssystems Tasks entweder als preemptive oder non-preemptive zu deklarieren: Nur für die als preemptive deklarierten Tasks muß Rechenzeit und RAM-Speicherbereich bereitgestellt werden, und nur bei den als non-preemptive deklarierten Tasks muß der Entwickler darauf achten, daß die Länge der Tasks nicht die Echtzeitbedingungen für das Steuerungssystem verletzt. Unter Inkaufnahme eines geringen Zusatzaufwands werden die Vorteile der beiden bekannten Betriebssysteme genutzt.

Patentansprüche

1. Steuergerät (1), insbesondere zum Steuern von Funktionen in Kraftfahrzeugen, das mit einem Prozessor (2), mit Datenspeichern (3, 4), mit Ein- und Ausgabeschaltungen (6) sowie mit einem Betriebssystem versehen ist, durch das die Abarbeitung von

Tasks zeitlich koordiniert und gesteuert wird, **dadurch gekennzeichnet,**

— daß das Betriebssystem eingerichtet ist sowohl für die Bearbeitung von Tasks nach einer Stapelverarbeitungsmethode (non-preemptive scheduling), bei der die Abarbeitung einer Task nicht unterbrochen werden kann, als auch für die Bearbeitung von Tasks nach einer Verdrängungsmethode (preemptive scheduling), bei der die Abarbeitung einer Task zum Abarbeiten einer anderen Task unterbrochen werden kann, und

— daß die Art der Abarbeitung jeder Task als Attribut zugeordnet ist, durch das die jeweilige Bearbeitungsmethode festgelegt wird.

2. Betriebssystem für ein Steuergerät (1), insbesondere zum Steuern von Funktionen in Kraftfahrzeugen, durch das die Abarbeitung von Tasks durch das Steuergerät zeitlich koordiniert und gesteuert wird, dadurch gekennzeichnet,

— daß Tasks bearbeitet werden sowohl nach einer Stapelverarbeitungsmethode (non-preemptive scheduling), bei der das Abarbeiten einer Task nicht unterbrochen werden kann, als auch nach einer Verdrängungsmethode, bei der das Abarbeiten einer Task zum Abarbeiten einer anderen Task unterbrochen werden kann (preemptive scheduling), und

— daß die Art der Verarbeitung jeder Task als Attribut zugeordnet wird, durch das die jeweilige Abarbeitungsmethode festgelegt wird.

Hierzu 3 Seite(n) Zeichnungen

– Leerseite –

Int. Cl.[6]: G 05 B 19/042
Offenlegungstag: 5. Oktober 1995

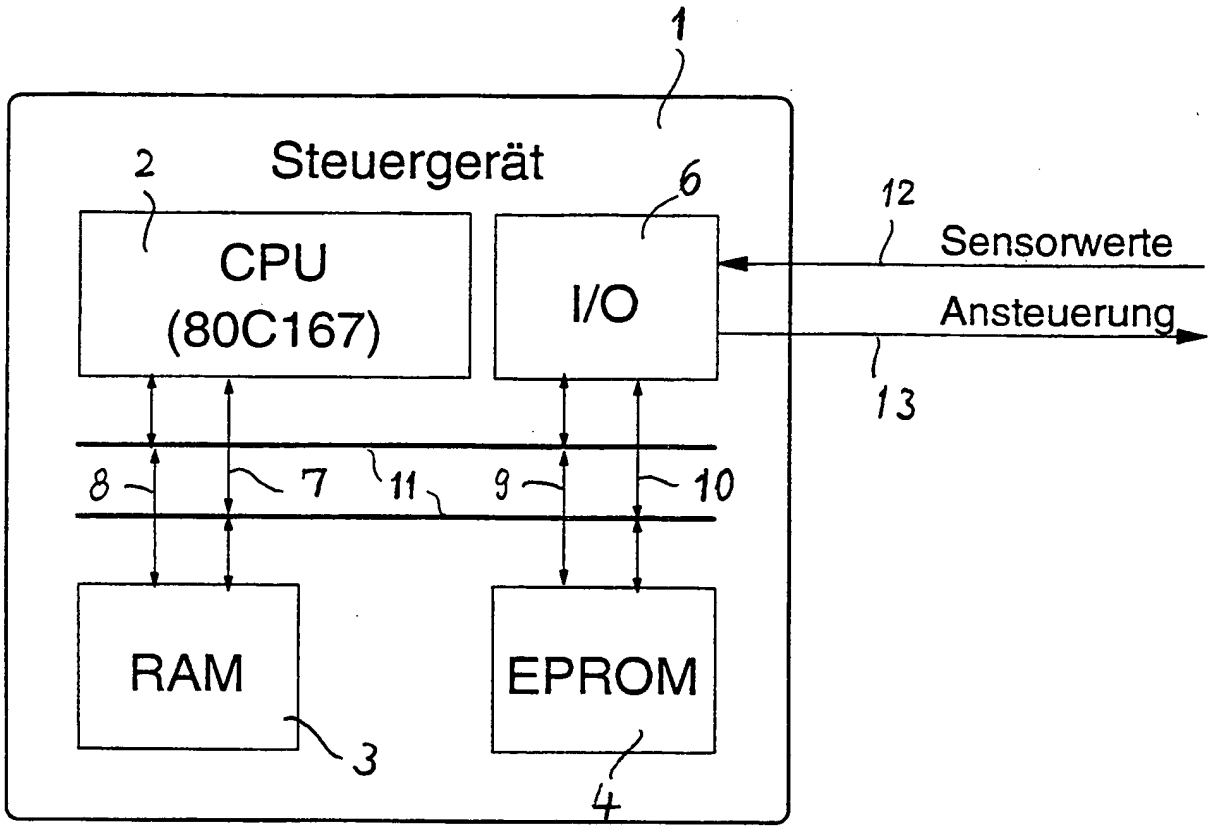


Fig. 1

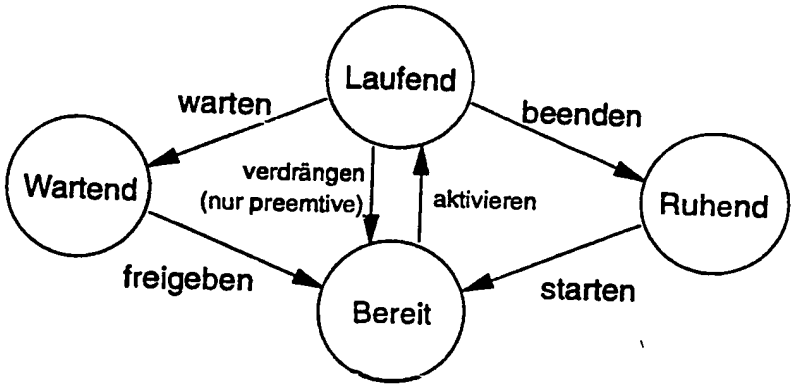

Laufend
warten
beenden
Wartend
verdrängen
(nur preemtive)
aktivieren
Ruhend
freigeben
starten
Bereit

Fig. 2

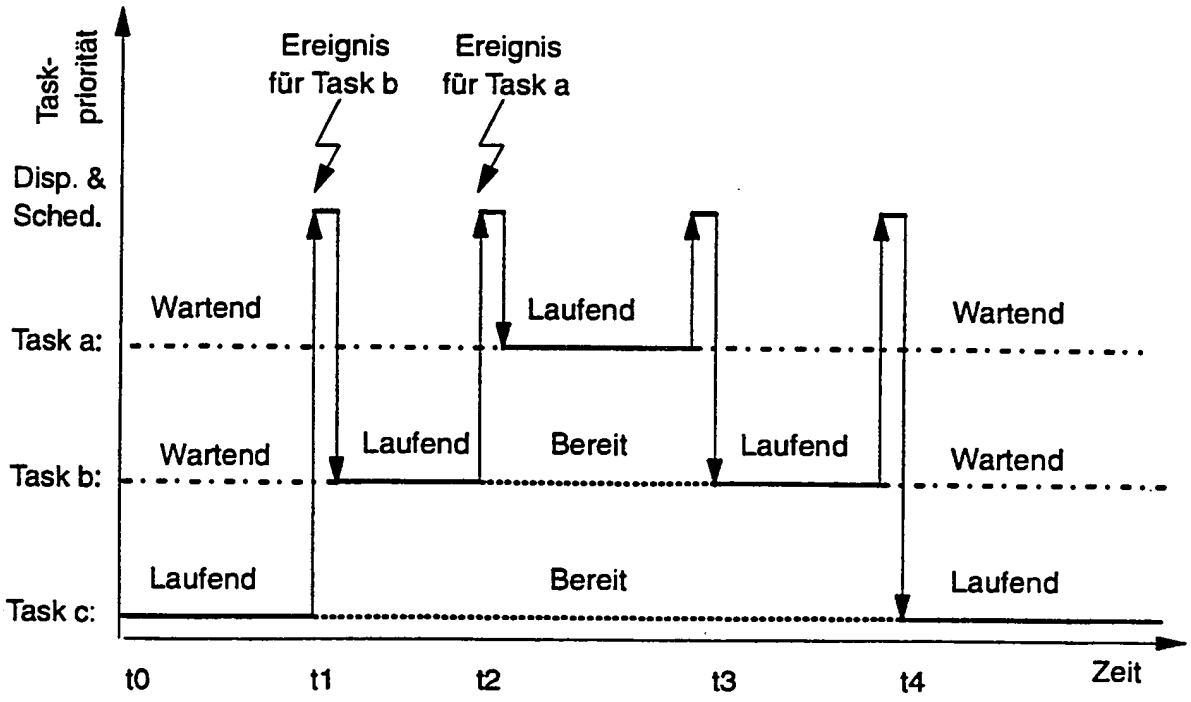

Task-
priorität
Disp. &
Sched.
Ereignis
für Task b
Ereignis
für Task a
Task a:
Wartend
Laufend
Wartend
Task b:
Wartend
Laufend
Bereit
Laufend
Wartend
Task c:
Laufend
Bereit
Laufend
t0
t1
t2
t3
t4
Zeit

Fig. 3

Nummer:
Int. Cl.[6]: G 05 B 19/042
Offenlegungstag: 5. Oktober 1995

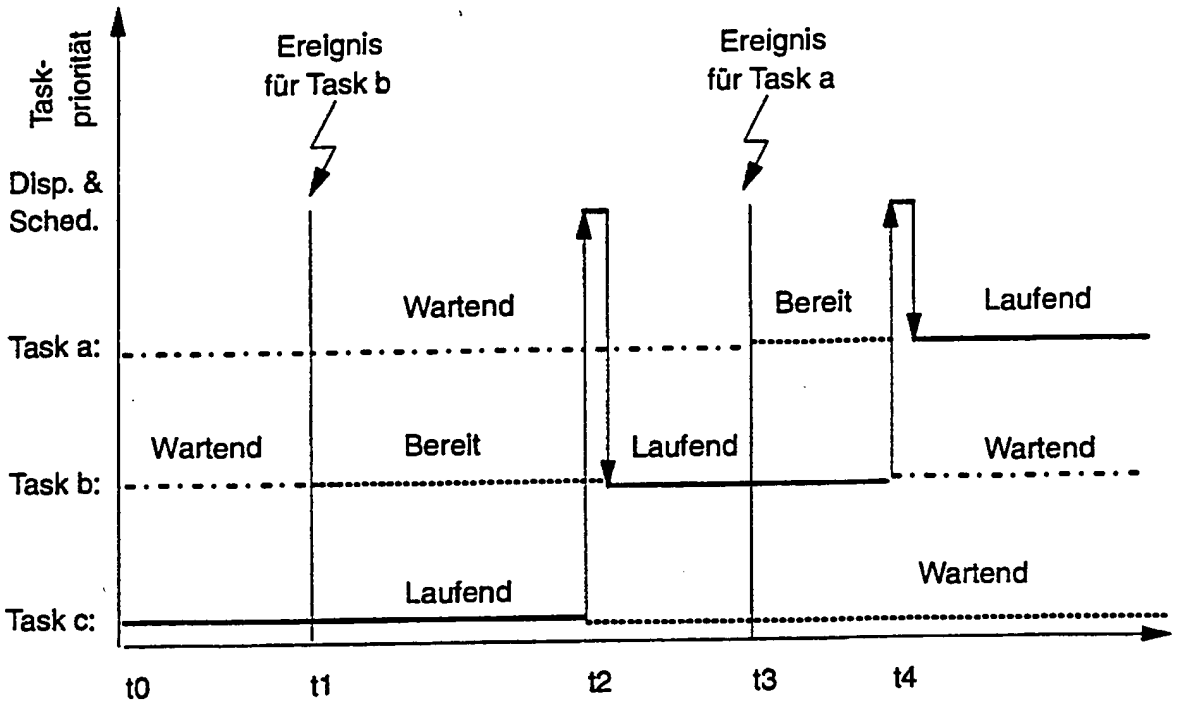


Fig. 4

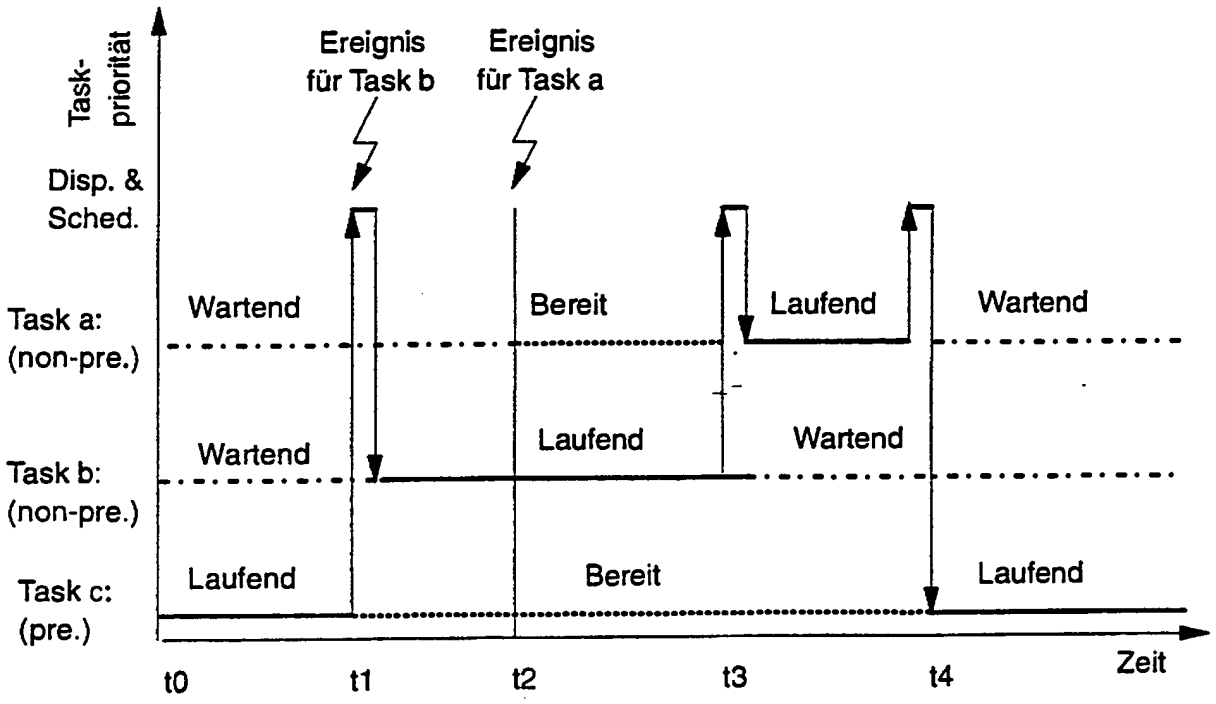


Fig. 5